Paving the Way towards Energy-Aware High Performance Computing

Manuel Dolz

manuel.dolz@informatik.uni-hamburg.de

October 22nd, 2014

Motivation

- High performance computing
  - Optimization of algorithms applied to solve complex problems
- Technological advance $\Rightarrow$ improve performance
  - Higher number of cores per socket (processor)
- Large number of processors and cores $\Rightarrow$ High energy consumption
- Tools to analyze performance and power to reduce energy consumption

Energy Efficient High Performance Computing

Motivation

- High performance computing
  - Optimization of algorithms applied to solve complex problems
- Technological advance $\Rightarrow$ improve performance
  - Higher number of cores per socket (processor)
- Large number of processors and cores $\Rightarrow$ High energy consumption
- Tools to analyze performance and power to reduce energy consumption

Energy Efficient High Performance Computing

Outline

Introduction

- **Parallel scientific applications**
  - Parallel apps., e.g., dense linear algebra: Cholesky, QR and LU factorizations
- **Tools for power and energy analysis**
  - Power-performance profiling/tracing tools, e.g., Extrae+Paraver

Environment to identify sources of power inefficiency

- **Approach**:
  - **Energy-aware techniques**: Leverage the available energy saving techniques: software and hardware.

Energy savings

Introduction

- **Parallel scientific applications**
  - Parallel apps., e.g., dense linear algebra: Cholesky, QR and LU factorizations
- **Tools for power and energy analysis**
  - Power-performance profiling/tracing tools, e.g., Extrae+Paraver

Environment to identify sources of power inefficiency

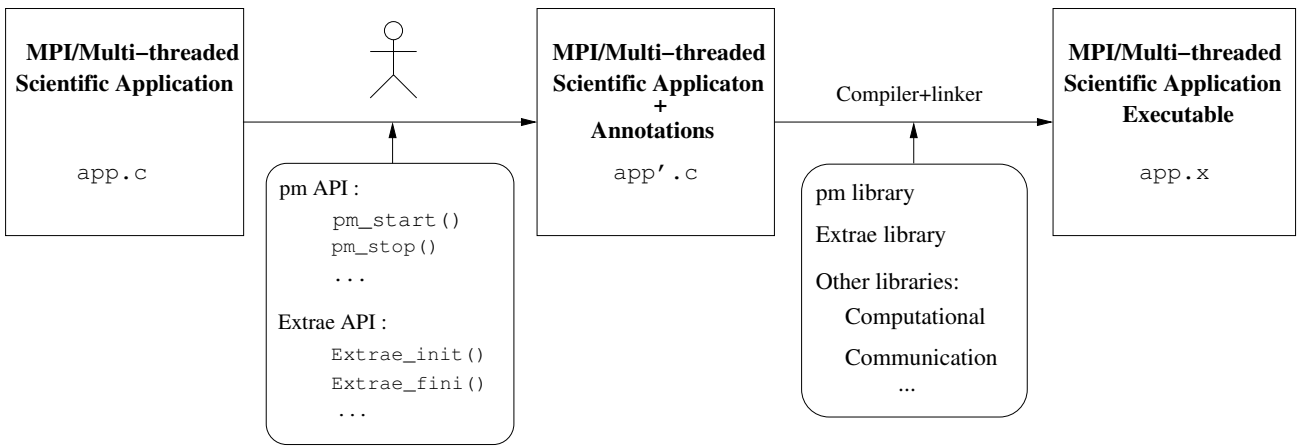
- **Approach**:
  - **Energy-aware techniques**: Leverage the available energy saving techniques: software and hardware.

Energy savings

Tools for performance and power tracing

Why traces?

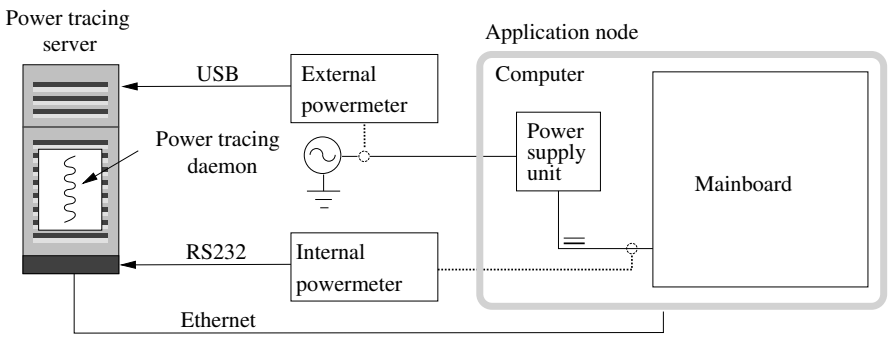
- Details and variability are important (along time, processors, etc.)
- Extremely useful to analyze performance of applications, also at power level!

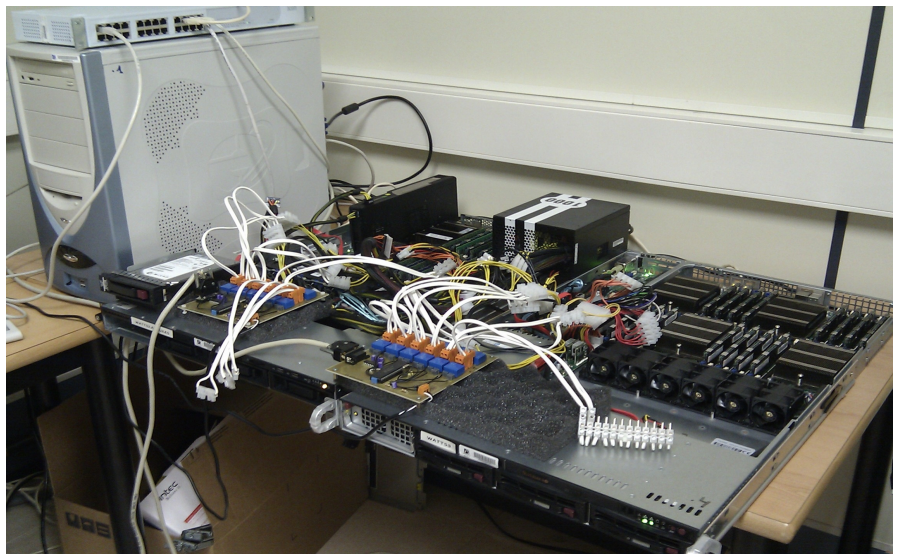


- Scientific application `app.c`
- Application with annotated code `app'.c`
- Executable code `app.x`

Performance and power measurement framework

- `Extrae+Paraver`: instrumentation and visualization tools from Barcelona Supercomputing Center (BSC)
- `pmlib` library:
  - Power measurement package of Jaume I University (Spain)
  - Interface to interact and use self-design and commercial power meters

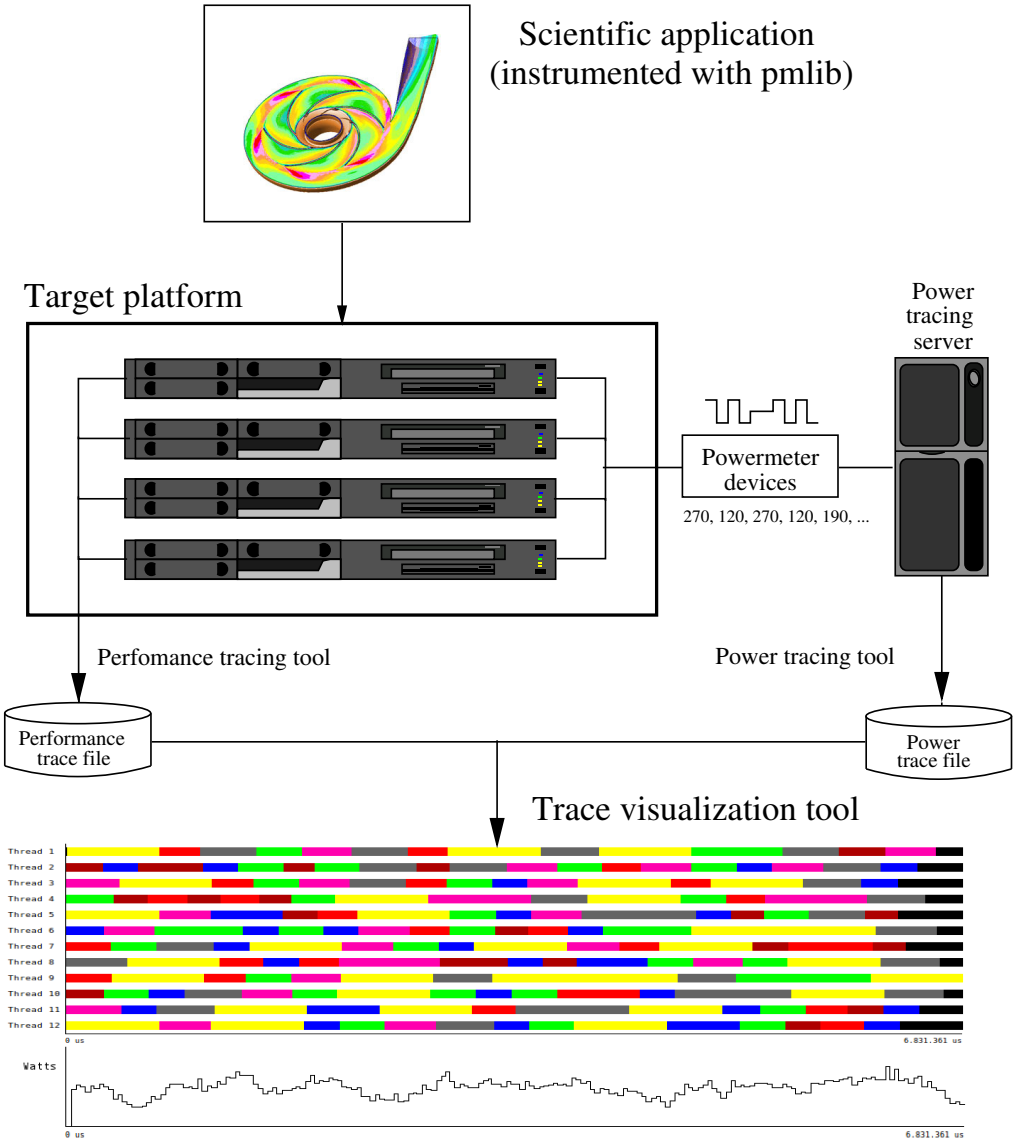




- **Server daemon**: collects data from power meters and send to clients
- **Client library**: enables communication with server and synchronizes with start-stop primitives

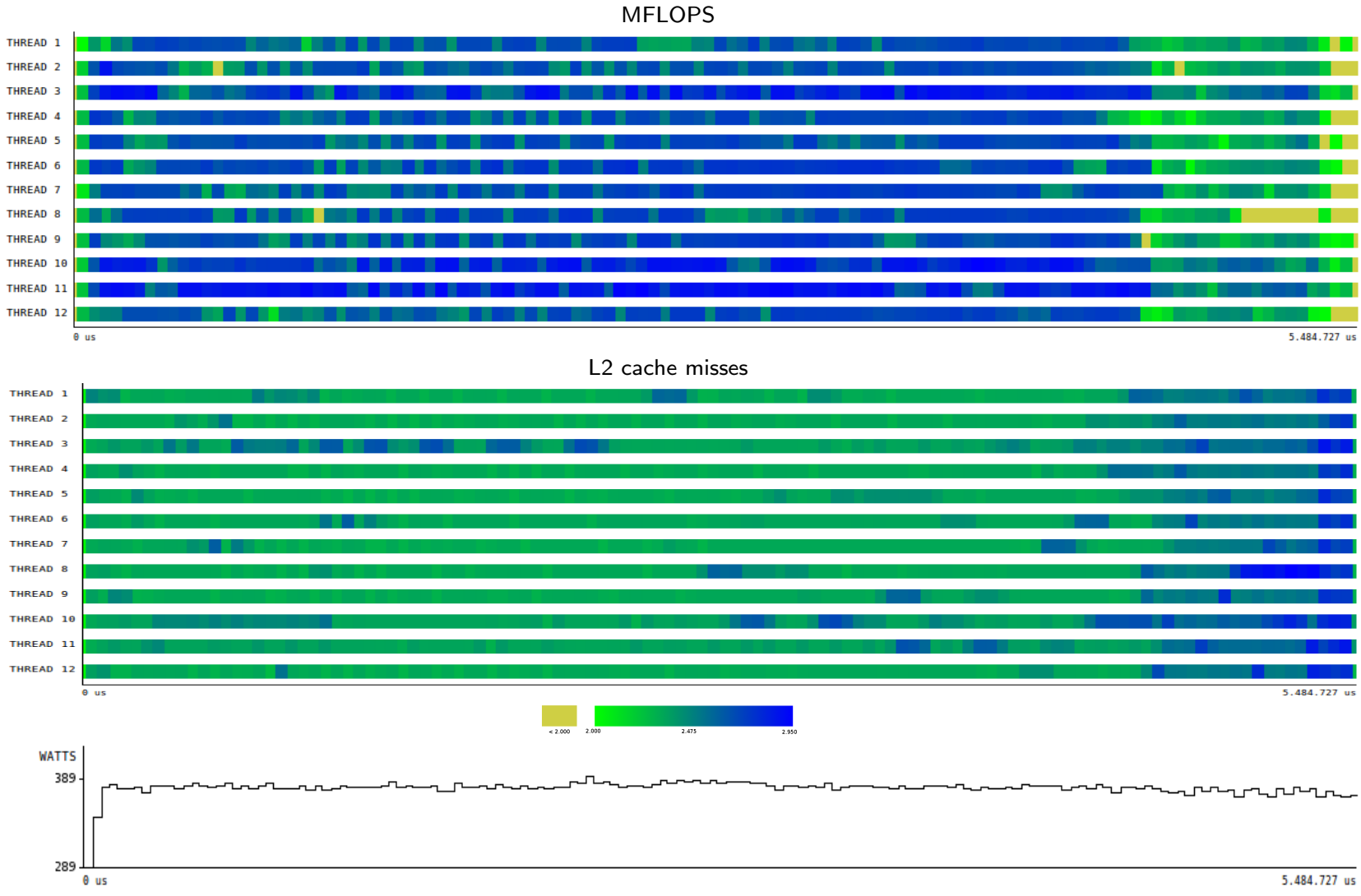
Code execution

Basic execution schema for tracing performance and power:

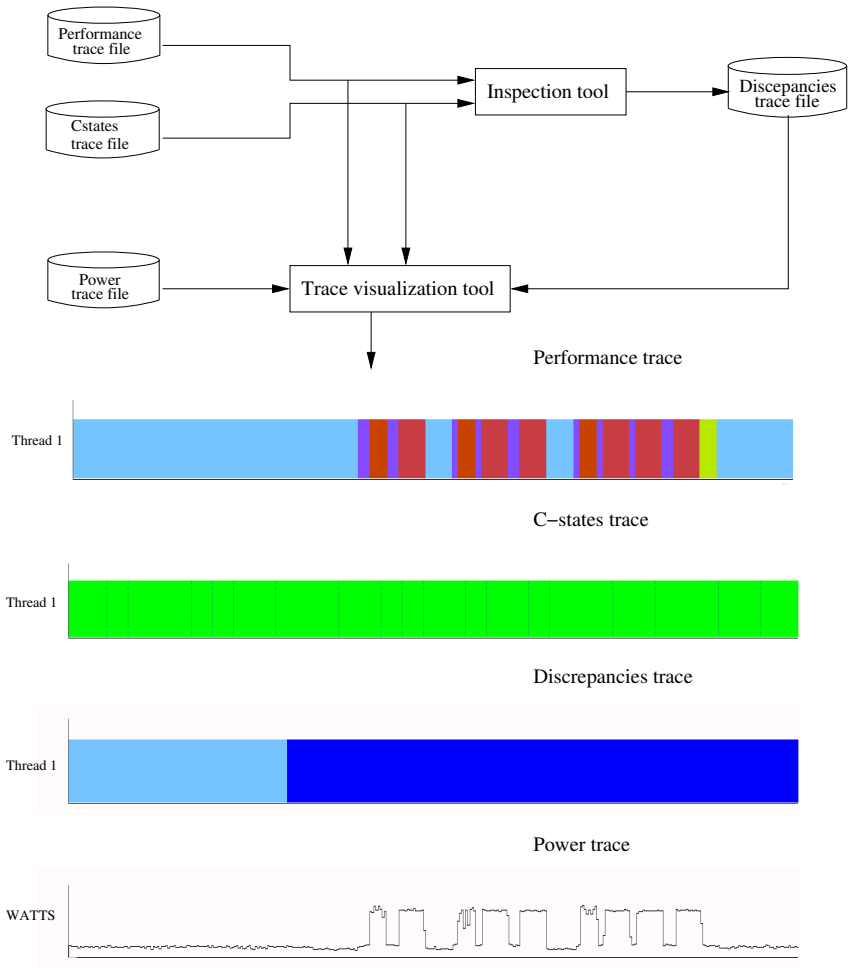


Example results

Cholesky factorization from MKL (`dpotrf`)

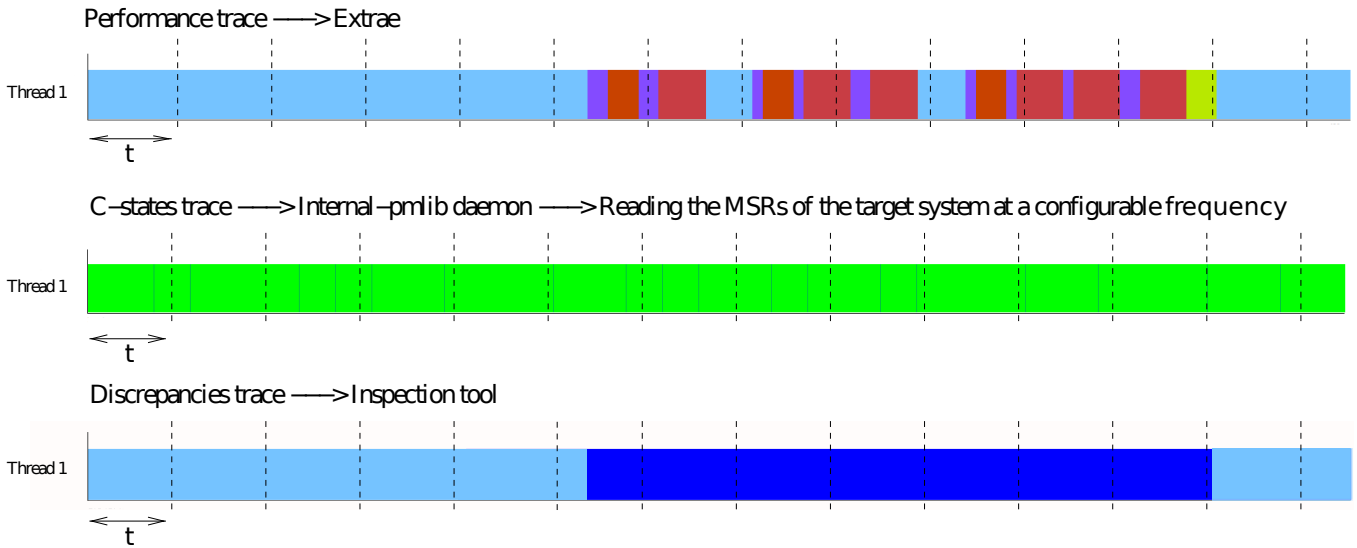


Parallel analyzer to detect power bottlenecks



- Automates and accelerates the inspection process
  - Comparison between the application performance trace and the C-states traces per core
- Flexible analyzer
- The user can define:
  - Task type that is “useful” work
  - Lenght of the analysis interval
  - Discrepancy threshold

Operation and implementation



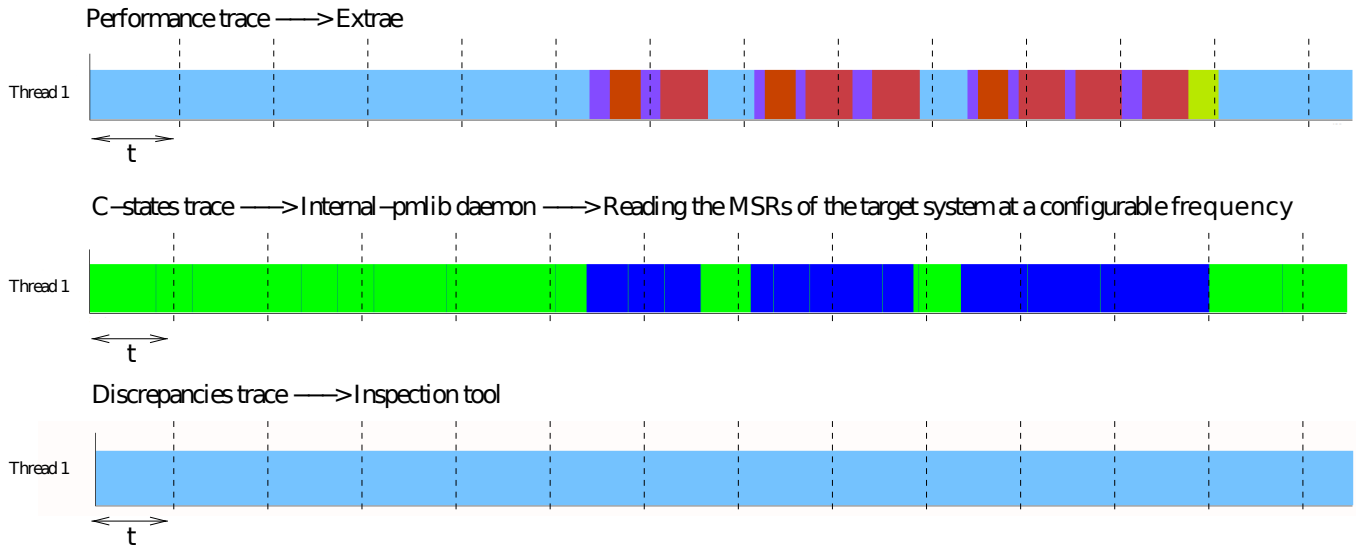
Implementation:

- Intervals of length t
- Python multithreaded analyzer

Result:

- Analytical $\Rightarrow$ (c, t_i, t_f, %divergence)
- Graphical $\Rightarrow$ Paraver

Operation and implementation



Implementation:

- Intervals of length t
- Python multithreaded analyzer

Result:

- Analytical $\Rightarrow$ (c, t_i, t_f, %divergence)
- Graphical $\Rightarrow$ Paraver

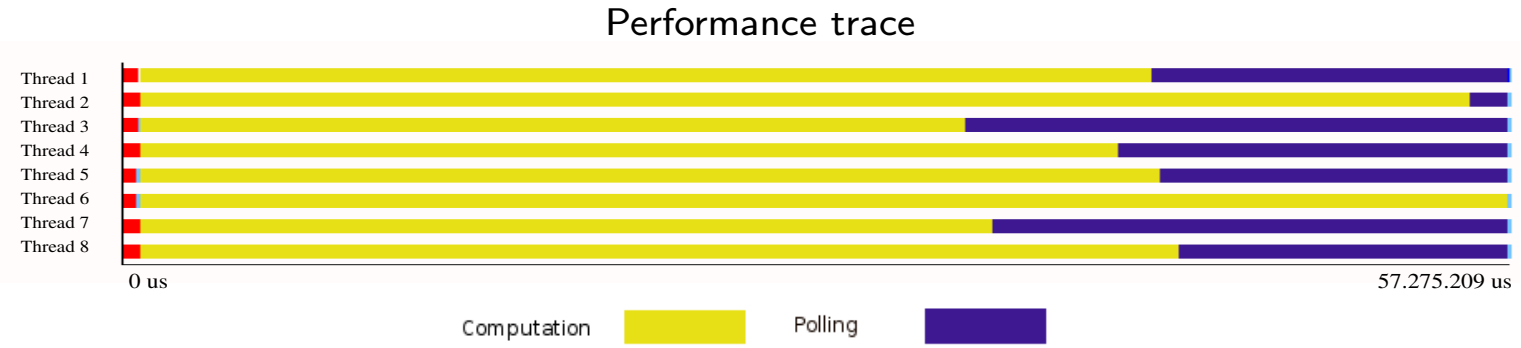
Example 1: ILUPACK

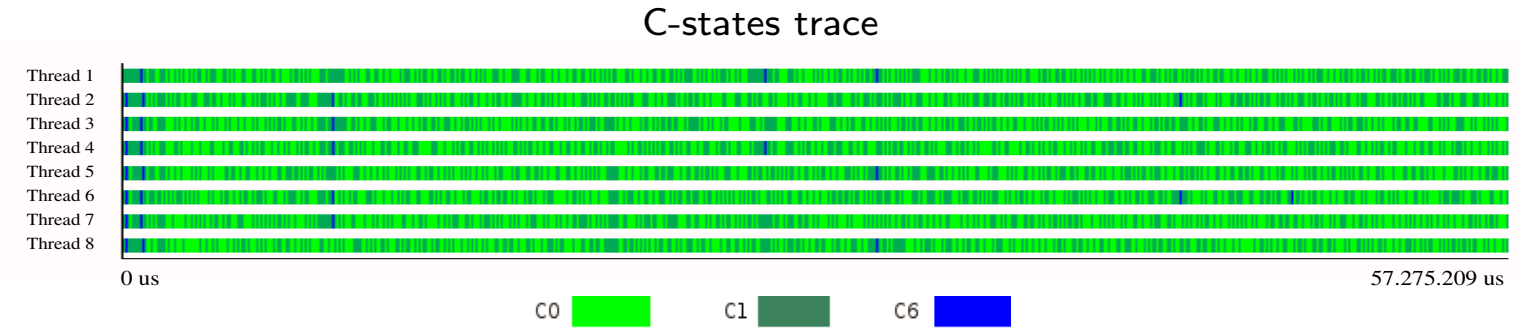
- ILUPACK: Concurrent solution of **sparse linear systems**
  - Multilevel preconditioners for general and Hermitian positive definite problems
  - **Parallelization**:

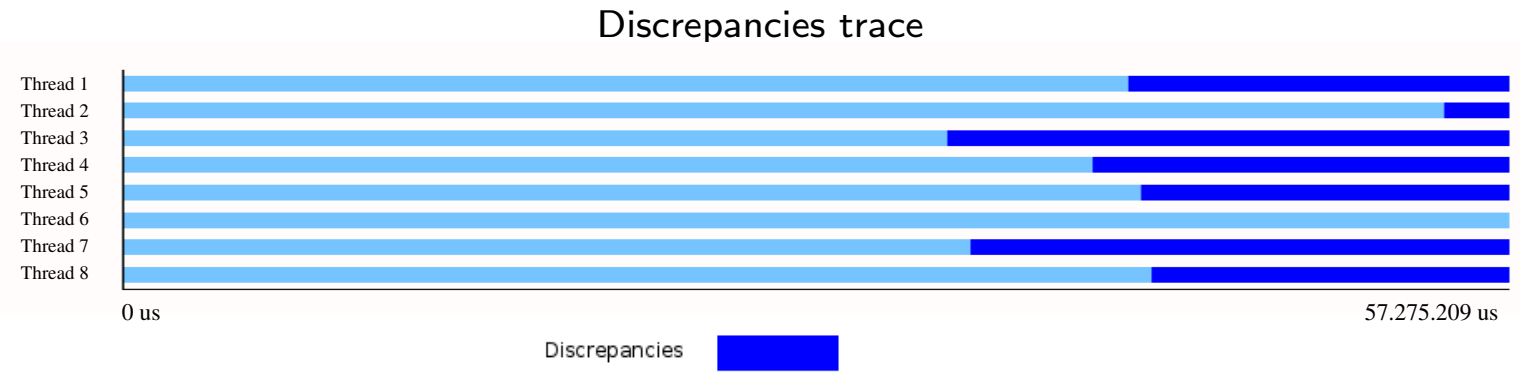
Task partitioning of the sparsity graph
$\Downarrow$
Task acyclic graph capturing dependencies
$\Downarrow$
Tasks mapped to threads on-demand at runtime

- Work by default: An idle thread **polls** the queue till a ready task becomes available
- **Test platform:**
  - Two Intel Xeon E5504 (4 cores, total of 8 cores) at 2.00 GHz
  - 32 GB of RAM
  - Linux Ubuntu 12.04

Example 1: ILUPACK





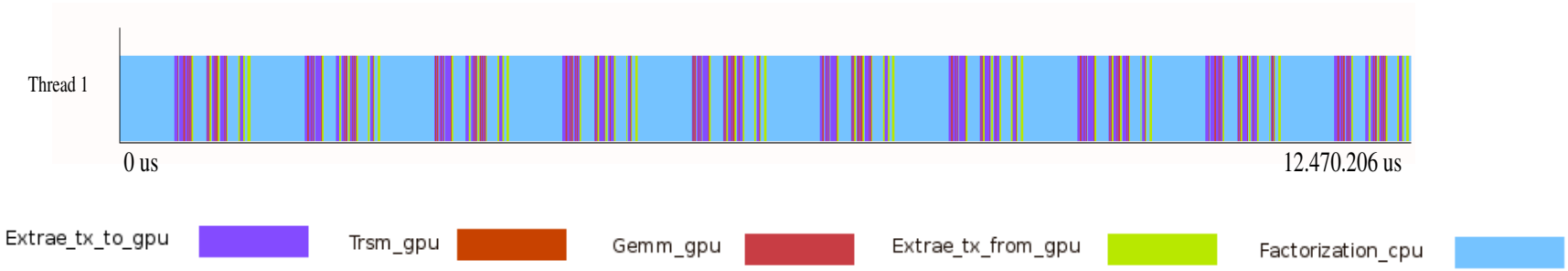


Example 2: LU Factorization

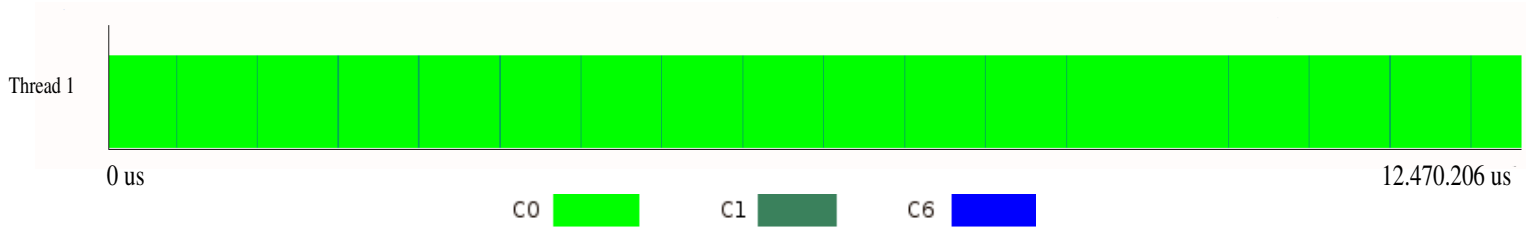
- LU factorization with partial pivoting of a **dense matrix**
- **FLA_LU** routine of **libflame** library
  - Parallelized with the **Supermatrix** runtime
- **Hybrid CPU–GPU computation:**
  - CPU $\Rightarrow$ Intel MKL
  - GPU $\Rightarrow$ NVIDIA CUBLAS
- Test platform CPU–GPU:
  - Intel Xeon i7-3770 with 16 GB of RAM
  - NVIDIA Tesla C2050 ("Fermi")

Example 2: LU Factorization

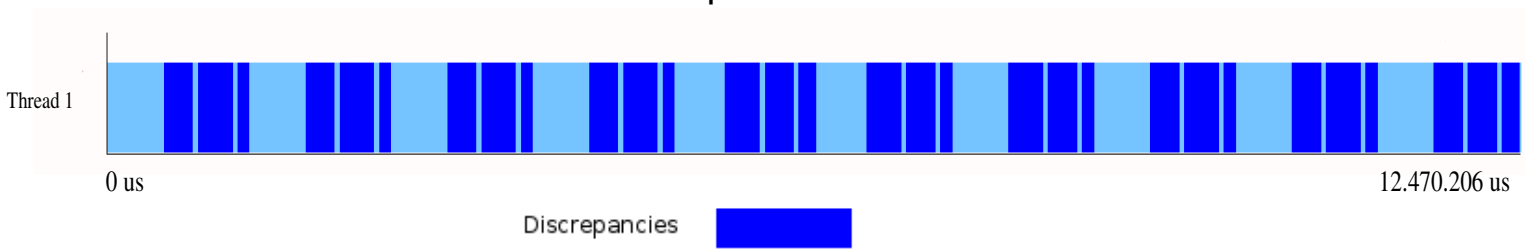
Performance trace



C-states trace



Discrepancies trace



Impact of power sinks

Statistical information for ILUPACK

	Computation	Polling	C0	C1	C6	Discrepancies
THREAD 1	72.00%	25.56%	99.33%	0.29%	0.39%	27.49%
THREAD 2	96.45%	2.50%	99.25%	0.26%	0.50%	4.77%
THREAD 3	59.90%	39.14%	99.53%	0.10%	0.37%	40.59%
THREAD 4	70.81%	28.13%	99.48%	0.10%	0.42%	30.11%
THREAD 5	74.00%	25.14%	99.29%	0.90%	0.61%	26.61%
THREAD 6	99.18%	0.00%	99.34%	0.22%	0.45%	0.00%
THREAD 7	61.52%	37.17%	99.53%	0.12%	0.35%	38.84%
THREAD 8	75.03%	23.69%	99.27%	0.10%	0.64%	25.74%

Estimation of the costs of the power sinks

- Time that cores are performing "useless" work $\Rightarrow$ Wasting power
- Potential savings:
  - (Power("guilty" core) - Power(power-saving state)) * total duration power sinks

How we can avoid power sinks?

Leverage HW energy-aware mechanisms

Outline

Energy-aware hardware techniques

ACPI (Advanced Configuration and Power Interface):
Industry-standard interfaces enabling OS-directed configuration, power/thermal management of platforms

Performance states (P-states):

- P_0: Highest performance and power
- P_i, $i > 0$: As i grows, more savings but lower performance

To DVFS or not? General concensus!

- Not for compute-intensive apps.: reducing frequency increases execution time linearly!
- Yes for memory-bounded apps. as cores are idle a significant fraction of the time.

But take care! $\Rightarrow$ In some platforms (AMD) reducing frequency via DVFS also reduces memory bandwidth proportionally!

P-states can be managed at socket level in Intel and at core level in AMD!

Energy-aware hardware techniques

ACPI (Advanced Configuration and Power Interface):
Industry-standard interfaces enabling OS-directed configuration, power/thermal management of platforms

Performance states (P-states):

- P_0: Highest performance and power
- P_i, $i > 0$: As i grows, more savings but lower performance

P-state P_i	VCC_i	f_i
P_0	1.23	2.00
P_1	1.17	1.50
P_2	1.12	1.20
P_3	1.09	1.00
P_4	1.06	0.80

Server AMD:
Two AMD Opteron 6128 cores @ 2.0 GHz (16 cores)

- $P = g\,(V^2\,f)$
- $E = \int_0^T P\,dt = g(V^2)$

$\longrightarrow$ DVFS!

To DVFS or not? General concensus!

- Not for compute-intensive apps.: reducing frequency increases execution time linearly!
- Yes for memory-bounded apps. as cores are idle a significant fraction of the time.

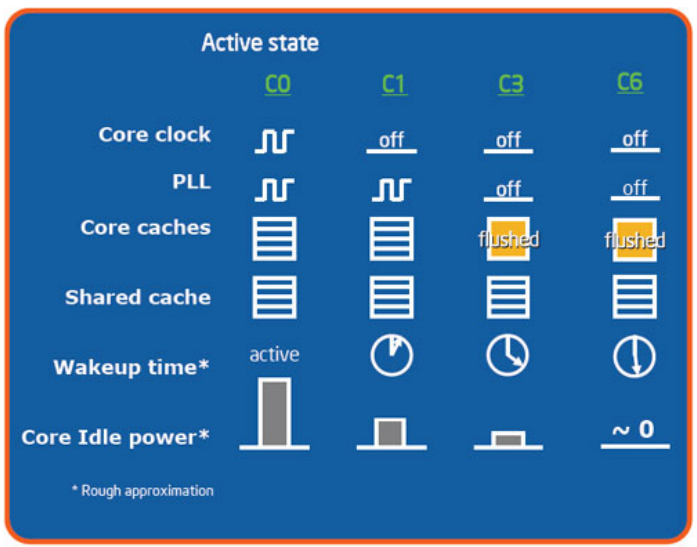
But take care! $\Rightarrow$ In some platforms (AMD) reducing frequency via DVFS also reduces memory bandwidth proportionally!

P-states can be managed at socket level in Intel and at core level in AMD!

Energy-saving states: P/C-states

Power states (C-states):

- C_0: normal execution (also a P-state)
- C_i, $i > 0$: no instructions being executed. As i grows, more savings but longer latency to reach C0

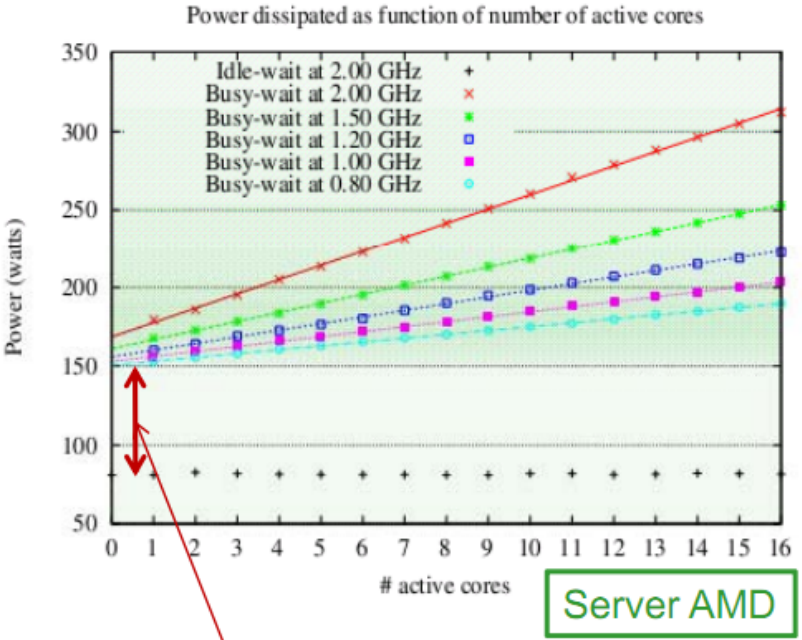


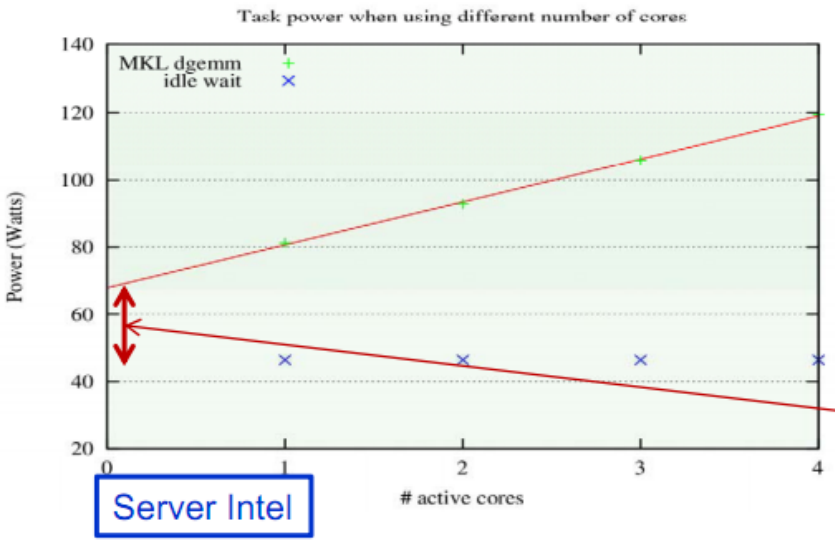
How to exploit C-states?

- Is impossible to change C-state at code level!
- Solution $\Rightarrow$ Set necessary conditions so that hw promotes cores to energy-saving C-states

Examples: P-states/C-states

- "Do nothing, efficiently..." (V. Pallipadi, A. Belay)
- "Doing nothing well" (D. E. Culler)





Opportunities to save energy via C-states!

Problem! Not straight-forward. No direct user control over C-states!

Energy-aware software techniques

Energy-aware techniques focused only on the "processors"!

Two approaches:

- *Slack Reduction Algorithm (SRA)*: Search for "slacks" (idle periods) in the DAG associated with the algorithm, and try to minimize them applying e.g. DVFS
- *Race-to-Idle (RIA)*: Complete execution as soon as possible by executing tasks of the algorithm at the highest frequency to "enjoy" longer inactive periods

Which is better?

- SRA: For memory-bounded apps., but take care of AMD platforms!
- RIA: For compute-intensive apps. like dense linear algebra algorithms

Energy-aware software techniques

Energy-aware techniques focused only on the "processors"!

Two approaches:

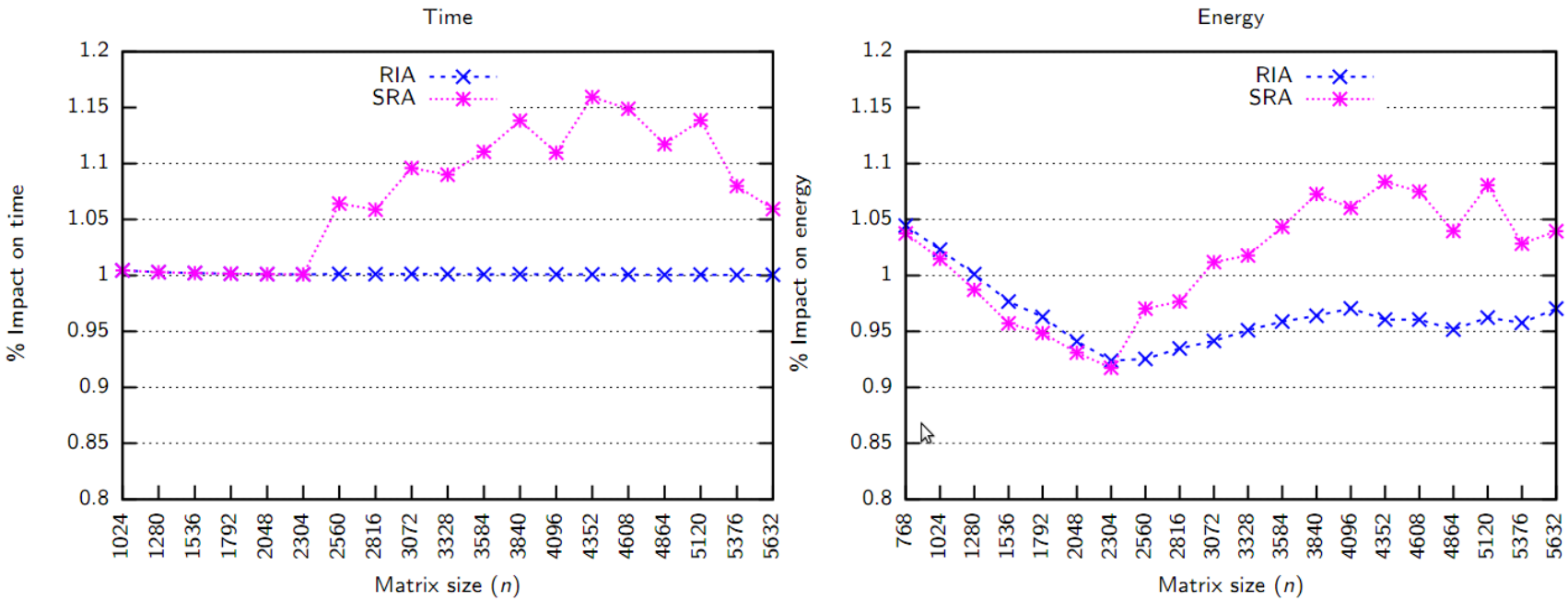
- *Slack Reduction Algorithm (SRA)*: Search for "slacks" (idle periods) in the DAG associated with the algorithm, and try to minimize them applying e.g. DVFS
- *Race-to-Idle (RIA)*: Complete execution as soon as possible by executing tasks of the algorithm at the highest frequency to "enjoy" longer inactive periods

Which is better?

- SRA: For memory-bounded apps., but take care of AMD platforms!
- RIA: For compute-intensive apps. like dense linear algebra algorithms

SRA vs. RIA

Impact of SRA/RIA on simulated time/energy for LUPP:

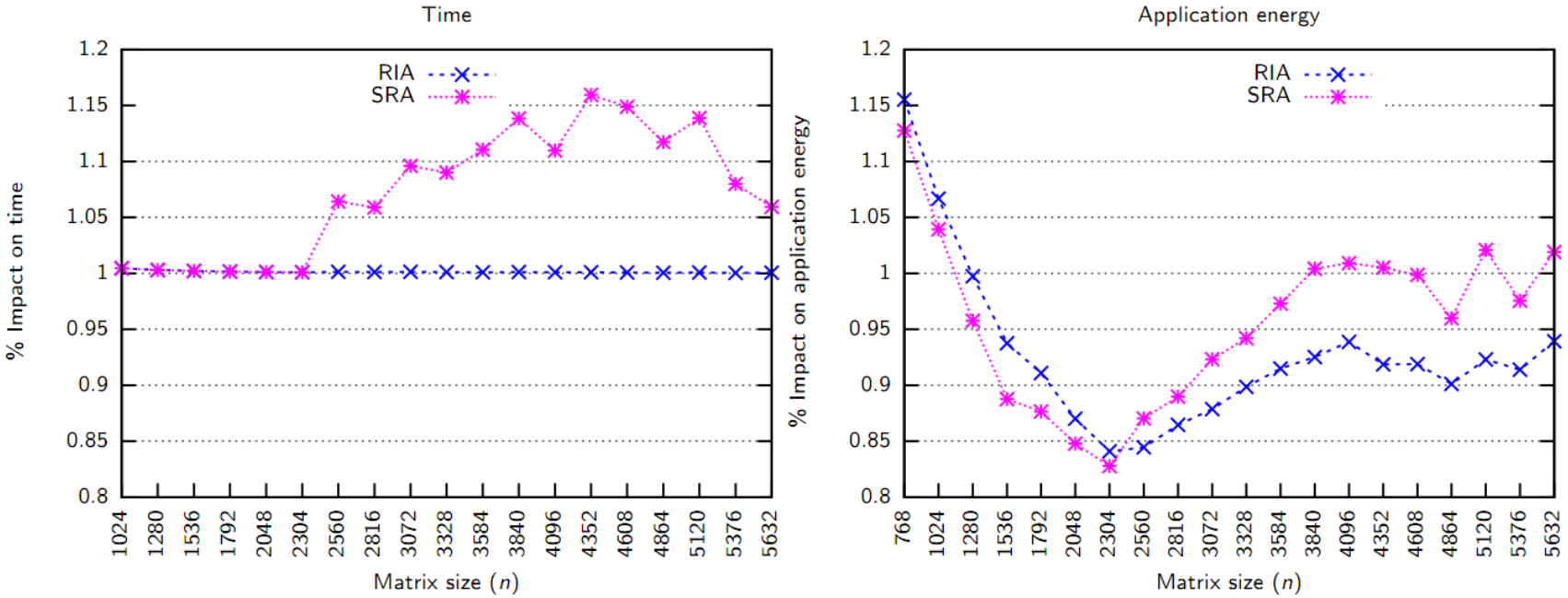


- **SRA**: Time is compromised, increasing the consumption for largest problem sizes
  - Increase in execution time due to SRA being oblivious to actual resources
- **RIA**: Time is not compromised and consumption is reduced for large problem sizes

SRA vs. RIA

Impact of SRA/RIA on simulated time/energy for LUPP:

only power/energy due to workload

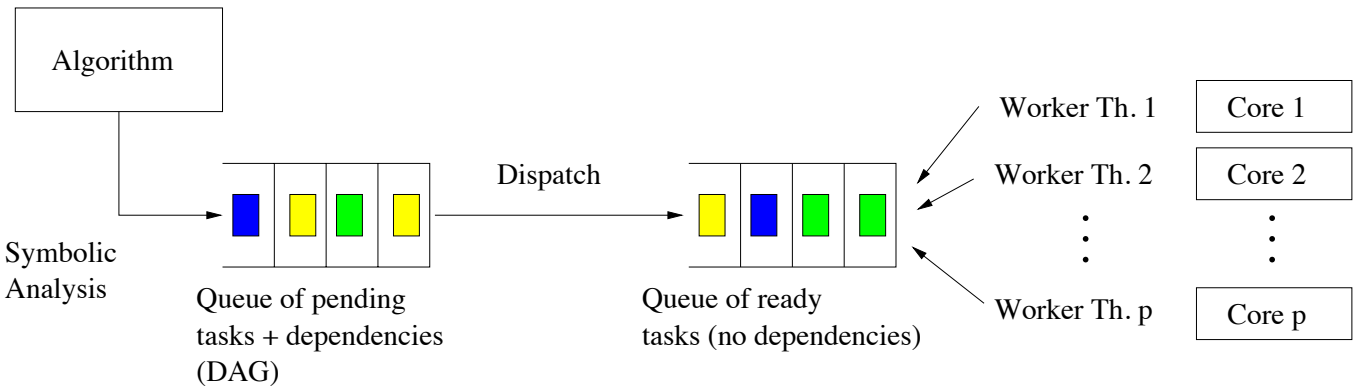


- **SRA**: Time is compromised, increasing the consumption for largest problem sizes
  - Increase in execution time due to SRA being oblivious to actual resources
- **RIA**: Time is not compromised and consumption is reduced for large problem sizes

Energy-aware software techniques

Dense linear algebra applications:

- Task-parallel execution of dense linear algebra algorithms: `libflame+SuperMatrix`



Problem:

- *Naive runtime*: Idle threads (one per core) continuously check the ready list for work Busy-wait or polling ⇒ Energy consumption!

Solution:

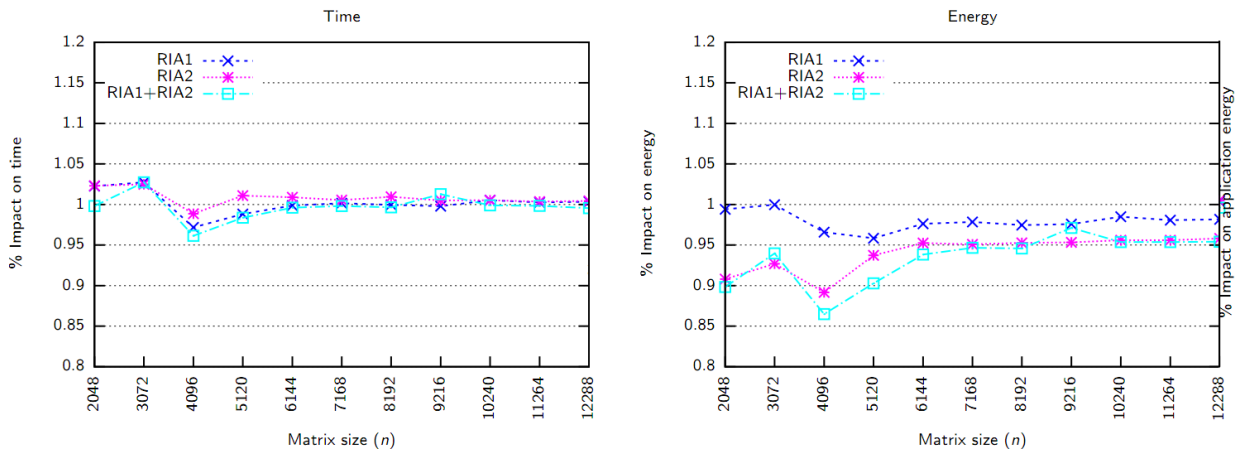
- *Race-to-idle*: Detect and replace "busy-waits" by "idle-waits": avoid idle processors doing polling!

Results: Dense linear algebra

Energy-aware techniques on **multicore** platforms:

- RIA1: Reduce operation frequency when there are no ready tasks: DVFS `ondemand` governor
- RIA2: Remove polling when there are no ready tasks (while ensuring a quick recovery): POSIX Semaphores

On multicore: `FLA_LU` (LUpp fact.) from libflame + SuperMatrix runtime



- Consistent savings around 5% for total energy and 7–8% for application energy
- Poor savings? Dense linear algebra operations exhibit little idle periods!

Results: Dense linear algebra

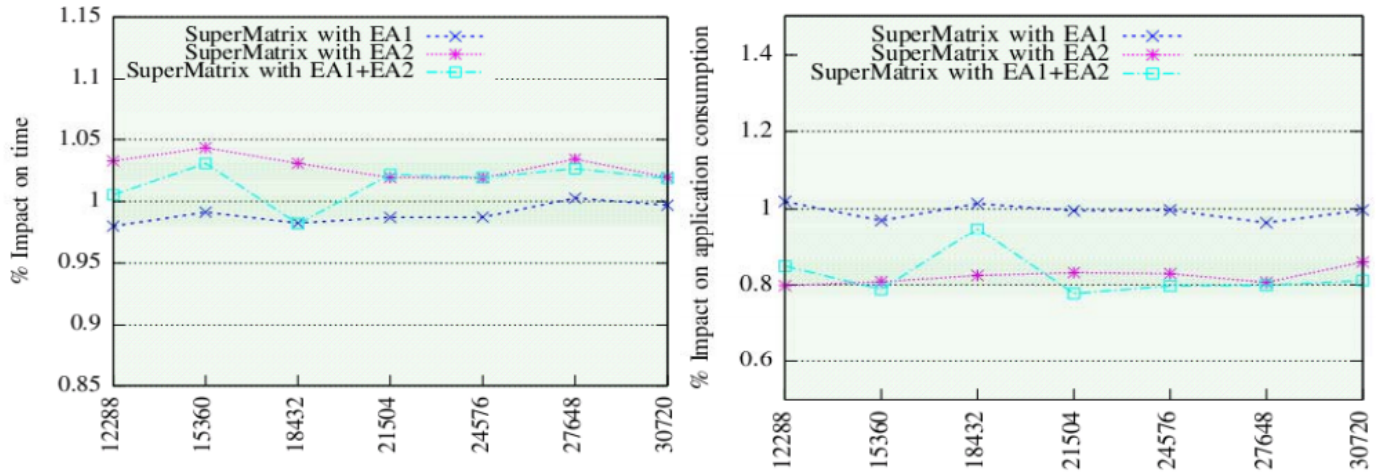
Why CPU+GPU (for some compute-intensive apps.)?

- High performance computational power / Affordable price / High FLOPS per watts ratio

Energy-aware techniques for **hybrid CPU+GPU** platforms:

- EA1: blocking for idle threads without task: POSIX Semaphores
- EA2: blocking for idle threads waiting for GPU task completion
  Set blocking operation mode (synchronous) for CUDA kernels

On hybrid CPU+GPU: `FLA_Chol` (Cholesky fact.) from libflame+SuperMatrix

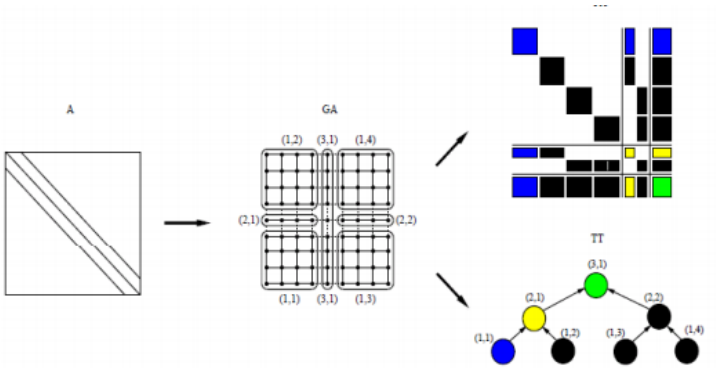


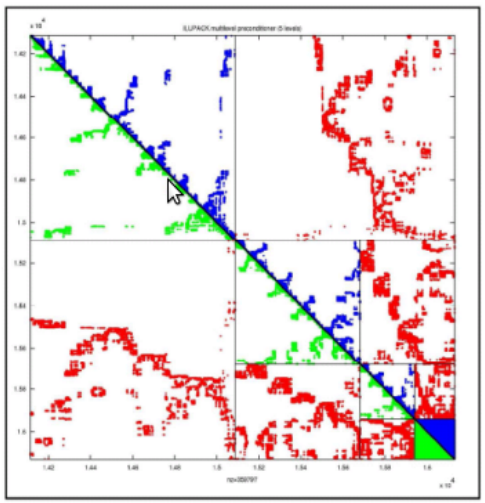
Execution of tasks in GPU makes CPU cores inactive during significant time!

Results: Sparse linear algebra

Sparse linear algebra applications:

- Task-parallel implementation of ILUPACK for multicore processors with *ad-hoc runtime*
- Sparse linear system from Laplacian eqn. in a 3D unit cube



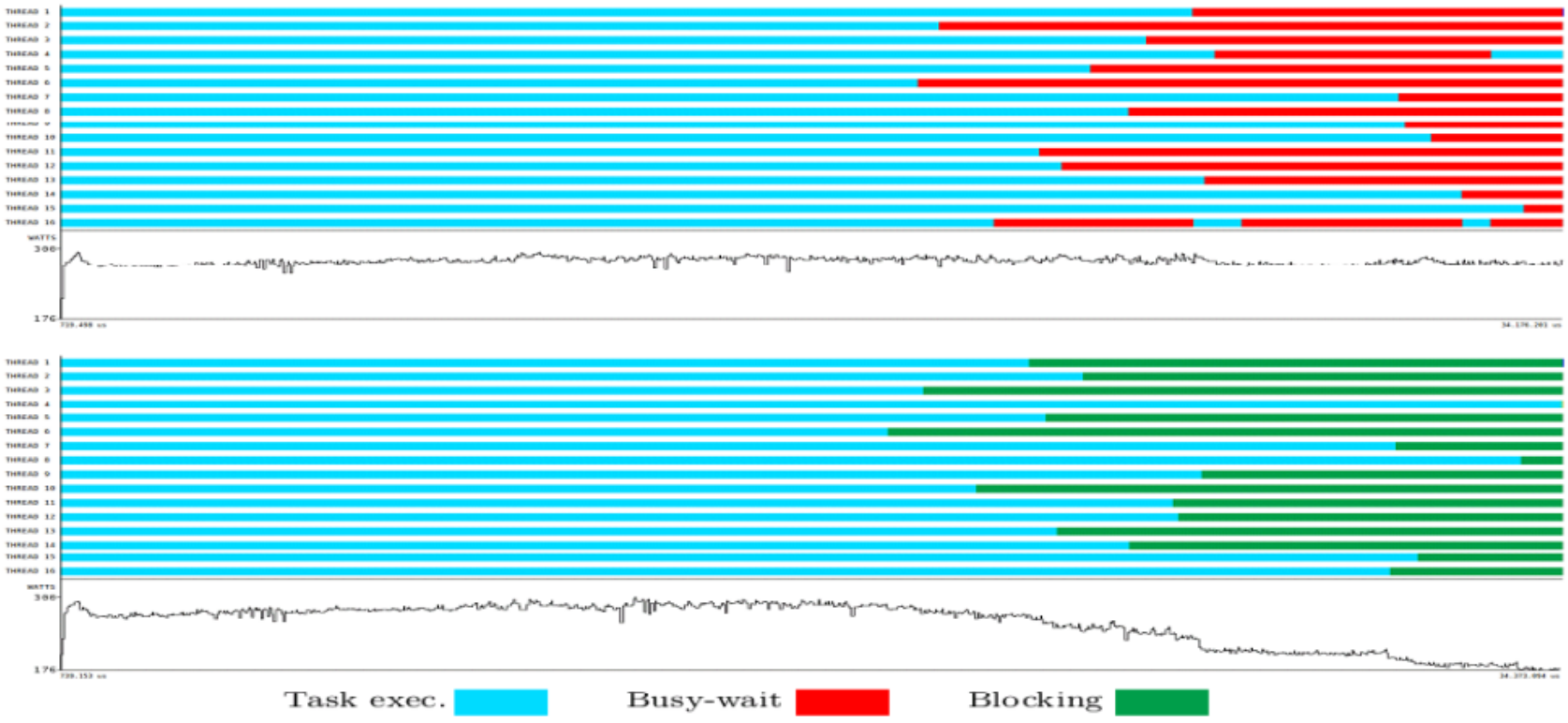


Energy-aware techniques:

- Application of RIA1+RIA2 techniques into *ad-hoc runtime*

Results: Sparse linear algebra

Polling vs. blocking for idle threads when obtaining ILU preconditioners:



Blocking vs polling for idle threads

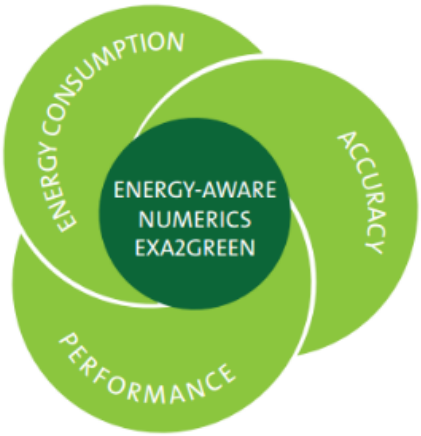
- Saving around 7% of total energy
- Negligible impact on execution time

...but take into account that

- Idle time: 23.70%, Dynamic power: 39.32%
- Upper bound of savings: $39.32 \cdot 0.2370 = 9.32\%$

Exa2Green Project

Exa2Green
energy-aware numerics



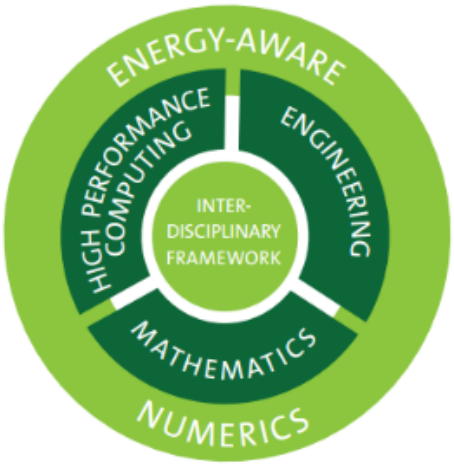
Energy-aware Sustainable Computing
on Future Technology
Paving the Road to Exascale Computing

www.exa2green.eu

Project Partners

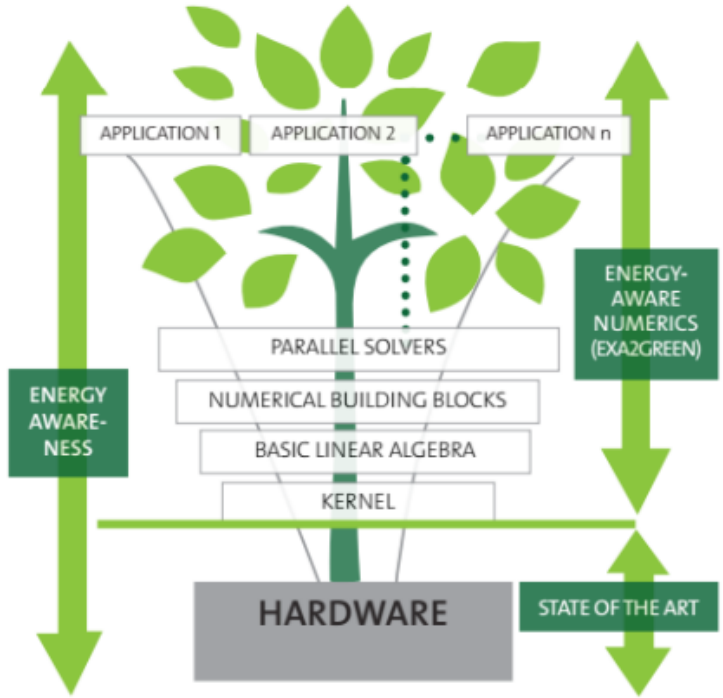
EMCL · Universitat Jaume I · IBM · KIT · Universität Hamburg · Universität Heidelberg · Steinbeis-Europa-Zentrum · ETH Zürich · CSCS

Exa2Green



Objectives

- Develop new multi-objective metrics for quantitative assessment and analysis of the energy profile of algorithms.
- Develop an advanced and detailed power consumption monitoring and profiling.
- Develop new smart algorithms using energy-efficient software models.
- Design a smart, power-aware scheduling technology for High Performance Clusters.
- Conduct a proof of concept using the COSMO-ART forecast model for aerosols and reactive trace gases.



Exa2Green is co-financed by the European Commission under the 7th Framework Programme

Conclusions

Tools for power/energy analysis

- Detect code inefficiencies in order to reduce energy consumption
- Automatic detection of power bottlenecks:

  Performance inefficiency $\Rightarrow$ hot spots in hardware and power sinks in code

Energy-aware software

- A battle to be won in the core arena
  - More concurrency
  - Heterogeneous designs
- A related battle to be won in the power arena
  - *"Do nothing, efficiently..."*, V. Pallipadi, A. Belay
  - *"Doing nothing well"*, D. E. Culler

Related publications

M. Barreda, M. F. Dolz, R. Mayo, E. S. Quintana-Ortí, R. Reyes
Binding Performance and Power of Dense Linear Algebra Operations
The 10th IEEE International Symposium on Parallel and Distributed Processing with Applications, 2012.

P. Alonso, R. M. Badia, J. Labarta, M. Barreda, M. F. Dolz, R. Mayo, E. S. Quintana-Ortí, R. Reyes
Tools for Power and Energy Analysis of Parallel Scientific Applications
The 41st International Conference on Parallel Processing, 2012.

M. Barreda, S. Catalán, M. F. Dolz, R. Mayo, E. S. Quintana-Ortí
Tracing the Power and Energy Consumption of the QR Factorization on Multicore Processors
12th International Conference on Computational and Mathematical Methods in Science and Engineering, 2012.

S. Barrachina, M. Barreda, S. Catalán, M. F. Dolz, R. Mayo, E. S. Quintana-Ortí
An Integrated Framework for Power-Performance Analysis of Parallel Scientific Workloads
The Third International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies, 2013.

María Barreda, Sandra Catalán, Manuel F. Dolz, Rafael Mayo, Enrique S. Quintana-Ortí
Automatic Detection of Power Bottlenecks in Parallel Scientific Applications
4th International Conference on Energy-Aware High Performance Computing, 2013.

P. Alonso, M. F. Dolz, F. D. Igual, R. Mayo, E. S. Quintana-Ortí
Reducing energy consumption of dense linear algebra operations on hybrid CPU-GPU platforms
The 10th IEEE International Symposium on Parallel and Distributed Processing with Applications, 2012.

Thanks for your attention!

Questions?